Indian Journal of Computer Science and Technology
https://www.doi.org/10.59256/indjcst.20260502014
Volume 5, Issue2 (May-August 2026), PP: 129-136.
www.indjcst.com
ISSN No: 2583-5300

An AI-Driven Quiz System using Multi-Agent Retrieval-Augmented Generation

Somay Singh[1], Rohitash Meena[2], Rajdeep Singh Hoon[3], Dr. Vivek Mehta[4]
[1,2,3]Department of Computer Science Engineering, Netaji Subhas University of Technology, Delhi, India.
[4]Supervisor, Department of Computer Science Engineering, NSUT, Delhi, India.

To Cite this Article: *Somay Singh[1], Rohitash Meena[2], Rajdeep Singh Hoon[3], Dr. Vivek Mehta[4], "An AI-Driven Quiz System using Multi-Agent Retrieval-Augmented Generation", Indian Journal of Computer Science and Technology, Volume 05, Issue 02 (May-August 2026), PP: 129-136.*

Abstract: *Classroom assessment continues to rely heavily on manually created quizzes, which consume a significant time from educators and they fail to reflect how students are actually progressing. General purpose AI tools that attempt to generate quiz questions tend to hallucinate may generate reasonable results but they are not grounded in terms of the material uploaded. This paper aims to addresses both problems by designing and implementing an AI-driven adaptive quiz system that combines Retrieval-Augmented Generation (RAG) with a multi-agent reasoning pipeline to produce assessments. The system implemented uses Retrieval-Augmented Generation to extract content from a teacher-uploaded PDF and generate multiple choice questions from it, making sure that the output stays grounded to the material. The generation pipeline is built along four agents built using LangGraph: an adaptive difficulty agent that adjusts the quiz difficulty based on the session average scores, a quiz generation agent that creates the questions, a review agent that checks each question for issues both through rule-based checks and an LLM evaluation, and an explanation agent that generates a short explanation for each correct answer. Human-in-the-loop interface was also introduced to let the instructor know that the flagged questions are marked in red with the reason and the correct are marked in green. The system was implemented was mainly built using Python, LangGraph, LangChain, FAISS, SentenceTransformers, Groq API, Streamlit, SQLite, and Plotly. It was also tested with 41 unit tests to verify that each component we implemented worked as we expected. When tested it across multiple sessions, the structural review passed about 92% of generated questions, quiz generation took between 18 and 35 seconds, and a clear difference in scores between easy and hard difficulty modes, which confirmed that the adaptive mechanism was actually working.*

Key Words: *Retrieval Augmented Generation (RAG), Agentic AI, Large Language Models, Multi-Agent System, Adaptive Assessment.*

I. INTRODUCTION

The rapid growth of digital learning resources has created both an opportunity and a challenge for educators. The digital content available made it easy to access them anytime but are also difficult to extract meaningful information from them without giving our considerable amount of time towards them. Even for making a simple assessment from a single unit requires the instructors to read the material thoroughly then select which will form the best questions and then finally converting it to the assessment.

Tools like Kahoot and ClassMarker solved the delivery problem but not the creation problem. They are dependent on pre-built question banks and they have no connection to what was actually uploaded as course material. Some tools have now adopted the use of AI in teaching context but as they have no idea to ground their LLM to the source material only they often face the problem of hallucination: LLM generates the output on the basis of the training data not the material it has been provided to be grounded to.

Another gap we noticed is that there is no concept to adapt the difficulty to improve student's experience. If a class struggled badly in the previous session, they still get the same difficulty level in the next quiz. There is no feedback loop that adjusts itself according to the students. So we decided to build a system that addresses all three of these issues together: generate questions from the actual teaching material, passes the checks and goes through HITL on the teacher's side before showing them to students, and adjust difficulty based on how the class did last time.

II. MOTIVATION AND BACKGROUND

The motivation for this project came from three specific problems we saw in existing tools. Mainly Need for intelligent and scalable asssessment, manually-intensive to set up different quizzes for different curriculum, generating questions from LLM may lead to hallucination and research gap.

A. Need for Intelligent and Scalable Assessment

The rapid digital transformation of education has increased the need for scalable and intelligent assessment methods. Although online platforms have improved access to learning resources, assessment systems often remain static and heavily dependent on manual effort. Traditional quiz-based evaluation fails to adapt to variations in classroom performance, which can

reduce student interest and learning effectiveness. As classrooms become more diverse in learning pace and understanding, there is a growing demand for assessment systems that can dynamically respond to collective performance trends in real time.

B. Manual-Intensive Assessment Preparation

Creating a quiz from scratch takes hours. It involves various tasks like reading the material, deciding what to test, analysing questions, making sure all the options are believable, not vague. Making sure the difficulty is appropriate for all. And these steps have to be performed repeatedly for different quizzes and different curriculum. If the system is able to automate the most basic steps, such that the repeated steps can be removed. Then it can save a significant amount of time and improve the quality.

C. Hallucination in Generative AI

Large language models (LLMs) are trained on huge amounts of data and gain their knowledge from it. But this does not help in the case when we want to extract knowledge from uploaded PDFs or any other document. It answers the user query from the data it has learnt, and when it is not able to generate the result for user query it provides factually incorrect data just to provide the output. This problem is called Hallucination. Without grounding the LLM to the source material, it cannot generate context-specific output for our uploaded documents. The Retrieval-Augmented Generation approach (RAG) helped us to ground our LLM to the provided material only and generate output for our specific document.

D. Research Gap and Proposed Direction

Despite these advancements, existing educational AI systems often treat content generation, performance analysis, and adaptive learning as separate processes. Few solutions provide a unified framework that combines adaptive quiz generation, session-based classroom assessment, and explainable AI feedback. This research addresses this gap by proposing an AI-driven adaptive quiz system that integrates multi-agent orchestration with retrieval-augmented knowledge processing. The goal is to enable scalable, intelligent, and data-informed assessment experiences suitable for modern classroom environments.

III. LITERATURE SURVEY

This section reviews the existing research that formed the foundation of our system. These studies helped us understand the existing limitations and provide with the novelty in existing systems.

3.1 Retrieval-Augmented Generation for Knowledge Intensive tasks

This is the paper that introduced the RAG framework to us and it is the most relevant to our work. The problem they solved was that LLM learned form the training on data if we asked something different to the model that it was not trained on then it gives output as something completely not related to what we asked or gives wrong answers confidently. The solution they provided was to use a retriever with the model it helps the model to first search the material provided and use it to generate the final output. This method outperformed the parametric based model and this was the core justification of using RAG instead of just writing to the model.

3.2 Autogen: Enabling Next-Gen Application via Multi-Agent Conversation

AutoGen is a framework from Microsoft Research that lets you build application where using multiple LLM as individual agents working together to complete a specific task. The idea they implemented was to give different tasks to different agents and breaking the complex task into simpler sub tasks.

The multi-agent design is also built on the same idea with having different implementation. AutoGen agents use conversational model to exchange message between them while our system uses LangGraph state to share the information between the agents. The system uses LangGraph instead of AutoGen is because is has sequential as well as conditional branches and the conditional branches are handled well in graph based architecture.

3.3 Survey of Retrieval-Augmented Generation for Large Language Models

This paper is a survey of how the RAG systems have evolved since the Lewis et al. paper. They classified RAG into three stages: Naïve RAG is the basic retriever then generator pipeline. Advance RAG is an improvement in terms of they re-rank the results and handle query better. Modular RAG are the most improved in terms that they allow us to input our individual components based on the use case. Reading this survey influenced the way to choose chunking strategy. Overlapping chunks were used as if the context is beyond a chunk if the same topic due to no overlapping it might get missed. All-MiniLM-L6-v2 was used because it is a semantic similarity based retriever not just a text based retriever. Our system falls under the Naïve RAG category as we use a retriever-generator chain.

3.4 A Systematic Review of Automatic Question Generation for Educational Purposes

This paper is a comprehensive review of the 93 papers published between 2015-2019 that focused on automatic question generation (AQG). This paper gave an insight about where it is currently stood, what method were used to tackle the problem and what gaps still remained. The main problem that it focused on was the time consumed while making manual questions for assessment. The existing techniques generated from texts they followed a template based approach. They listed that still systems were not addressing the feedback problem telling why the option is correct or incorrect. They listed that AQG is still growing and significant work need to be done in adapting the difficulty of the questions and feedback generation.

The system addressed the issues of feedback generation and adaptive difficulty. The feedback was presented on both the student

(Explanation) as well as teacher side (HITL). Adaptive difficulty was implemented as the first step in quiz generation that adapts as per the score in the session.

3.5 Comparative Analysis of RAG, Fine-Tuning & Prompt Engineering

This paper compared three ways of making the LLM work for specific domains – RAG, Fine-tuning and prompt engineering. They tested each of them in chatbot development. The findings stated that Fine-tuning gave the highest accuracy of 87.8% but required labelled training data, GPU and needed to be done everytime the data changes. Prompt engineering is the simplest and most economical but can limit the output in factual data. RAG is in the middle of both it does not require training everytime the data changes. This paper helped identify that RAG is the most practical choice for the system as it will address changing data source everytime.

3.6 Intelligent tutoring systems

This paper demonstrated that intelligent tutoring systems can achieve learning gains comparable to human tutoring under certain conditions. This work underscores the importance of adaptive assessment mechanisms in enhancing educational outcomes. However, many classical ITS implementations rely on rule-based adaptation and lack scalability for modern AI-driven classroom environments.

IV. SYSTEM ARCHITECTURE

The proposed AI-driven adaptive quiz system integrates document-grounded content generation, adaptive reasoning mechanisms, and real-time performance analytics to support scalable and context-aware evaluation workflows. The architecture is organized into interconnected modules that collectively enable intelligent quiz generation, classroom interaction, and adaptive feedback.

A. Teacher Interaction Module

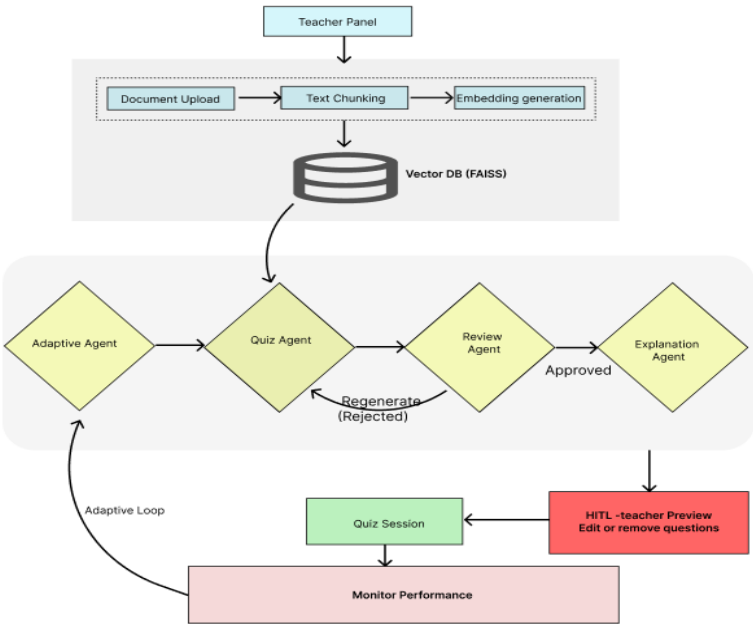

Figure 1 Teacher's Panel Workflow

We built the teacher panel with the help of Streamlit. It has a PDF uploader, a text box for the topic focus, an optional field for a previous session code, and a slider for number of questions (3 to 10). After the quiz is generated and reviewed through the HITL interface the teacher can edit the flagged questions or can remove them completely. After the corrections have been made the teacher can generate a session code for the students to attempt the quiz.

There is also a separate section where the teacher can load analytics. They just have to enter the session code for the quiz sessions and then can view the report of the student's marks.

B. Student Interaction Module

The student panel is a separate navigation option in Streamlit. The student types their name and session code, and the quiz loads from the database. The questions are in the form of radio button quiz. They answer each question submit when done. The system then calculates the score, stores the attempt and shows a detailed feedback (explanation) with every question. The chosen answer, the correct answer whether it was right or wrong and the LLM-generated explanation that helps the student's to learn from the mistakes and is not there just to provide them explanations.

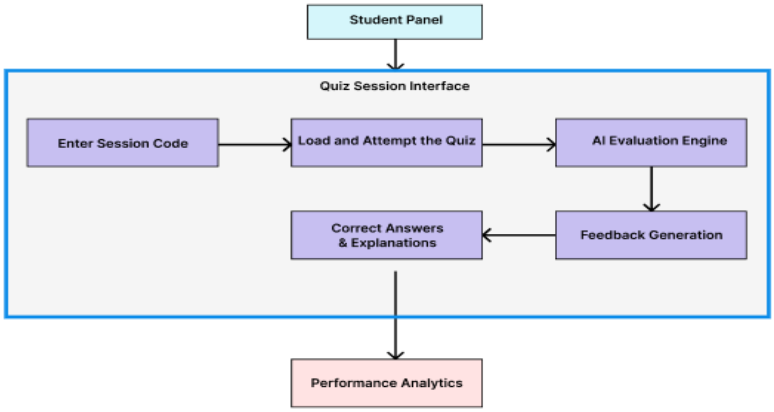

Figure 2 Student's Panel workflow

C. Retrieval-Augmented Knowledge Processing Layer

At the core of the system lies a Retrieval-Augmented Generation (RAG) pipeline that grounds the LLM assisted quiz generation to the teacher provided resources. Uploaded documents are processed through embedding models and transformed into vector embeddings. These embeddings are stored within a vector database with similarity search framework, enabling efficient retrieval of relevant knowledge during quiz generation. This mechanism improves factual accuracy and contextual relevance in generated content.

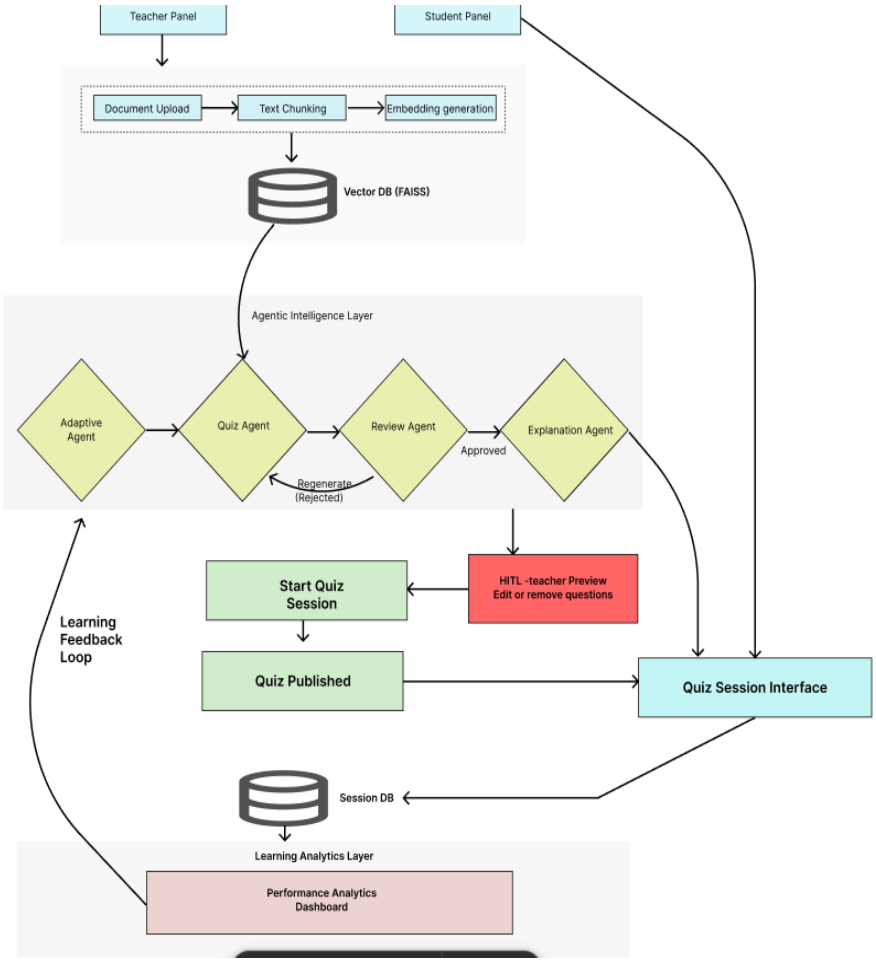

Figure 3 Proposed Overall System Architecture

D. Human-in-the-Loop Architecture

After the review agent has completed its specific task the teacher sees every question in an editable format. Questions that have passed are shown in green border. Questions the reviewer flagged show a red border with the reason and suggested fix right there in the interface. The teacher can edit the question text, change any of the four options, pick a different correct answer from a

dropdown, or even delete the question. When satisfied, the teacher clicks confirm and the session gets created.

There is also a regenerate button that clears the current quiz and lets the teacher start generation again from scratch. After generating the session code the teacher can then share the code with the student's to login and attempt the quiz.

E. Multi-Agent Reasoning Framework

The multiple-agent pipeline is built using LangGraph. All the four agents share a single state such that they can conversate with each other and update the state as per their function.

The graph has four sequential nodes. At the review agent a conditional branch is implemented if the quiz has been passed by the reviewer then it proceeds to the explanation agent. If it has been rejected then it for once goes back to the quiz generation agent to again generate the quiz. If it has already been rejected once, it routes to the end node and stops executing. This prevents the pipeline from getting stuck in an infinite loop while still giving the system one chance to fix a bad generation.

F. Performance Analytics and Adaptive Feedback Module

To enable adaptive intelligence, the system incorporates a performance analytics module that monitors student responses and derives session-level learning indicators. These analytics guide dynamic modification of quiz difficulty across successive assessment cycles, forming a closed-loop feedback mechanism. By aligning assessment complexity with observed classroom performance trends, the system enhances both learning effectiveness and evaluation relevance.

Overall, the architecture combines retrieval-grounded content generation, agent-based reasoning, and adaptive analytics to form a scalable intelligent tutoring infrastructure. Its modular design supports flexible deployment across diverse educational settings while maintaining transparency, adaptability, and operational efficiency.

V. METHODOLOGY AND IMPLEMENTATION

The proposed adaptive quiz system follows a structured multi-stage methodology that integrates Retrieval -Augmented Knowledge Processing Layer, Multi Agent Framework, Human-in-the-Loop, Database Design and Testing Framework. The methodological framework is designed to enable scalable quiz generation and dynamic difficulty modulation within classroom environments.

A. Retrieval-Augmented Knowledge Processing Layer

The knowledge retrieval pipeline underwent iterative refinement during system development. An initial implementation utilised ChromaDB as the vector store, selected for its native compatibility with LangChain and support for persistent storage. However, empirical evaluation against realistic instructional documents revealed performance limitations that prompted a transition to FAISS for the final system. FAISS offers superior in-memory similarity search efficiency at the document scales encountered in classroom deployments and eliminates the requirement for a dedicated server process, thereby reducing operational complexity and improving cross-platform portability.

The retrieval-augmented generation (RAG) pipeline is structured across four sequential stages. In the first stage, uploaded PDF documents are parsed using PyPDFLoader, which extracts textual content on a page-by-page basis, yielding structured output comprising both content and associated metadata. In the second stage, the extracted text is segmented into overlapping chunks using Recursive Character Text Splitter. This splitter was selected over alternatives due to its capacity to preserve semantic continuity across paragraph boundaries and multi-line constructs. In the third stage, each text chunk is encoded into a dense vector representation using the all-MiniLM-L6-v2 SentenceTransformer model. This model was chosen for its favourable balance between semantic embedding quality and computational efficiency, enabling deployment on CPU-based hardware without GPU dependency. In the fourth stage, the resulting embeddings are indexed within a FAISS retrieval structure. At inference time, the instructor-specified topic is embedded using the same model, and the most semantically similar document chunks are retrieved and forwarded to the quiz generation agent as contextual grounding.

During implementation, a memory inefficiency was identified wherein both the embedding storage module and the retriever module independently instantiated their own copies of the Sentence Transformer model, resulting in duplicate loading of the approximately 90 MB model into system memory. This issue was resolved by defining the model as a shared module-level variable, ensuring a single instantiation is reused across both components.

B. Multi-Agent Framework

The multi-agent assessment pipeline is implemented using LangGraph, a graph-based orchestration framework for constructing stateful agent workflows. All four agents operate over a shared state object, enabling inter-agent communication and incremental state updates as execution progresses through the pipeline.

The pipeline is structured as a directed graph comprising four sequential processing nodes, with a conditional branching mechanism applied at the review agent stage. If the generated quiz passes the review agent's quality evaluation, execution proceeds to the explanation agent. If the quiz is rejected, it is routed back to the quiz generation agent for a single regeneration attempt. If a second rejection occurs, execution is directed to the terminal node, halting further processing. This bounded retry mechanism prevents unbounded looping while affording the system one corrective iteration in the event of an initial generation failure.

1. Adaptive Difficulty Agent: The adaptive difficulty agent constitutes the first processing stage in the multi-agent pipeline. Difficulty modulation is governed by a threshold-based algorithm that evaluates the mean student score from the preceding assessment session. Specifically, if the session average falls below 0.4, the quiz is classified as having been excessively difficult,

and a simplified difficulty strategy is applied to the subsequent session. If the average lies between 0.4 and 0.7, the quiz is deemed to be of moderate difficulty, and a balanced standard difficulty is maintained. An average exceeding 0.7 indicates that the quiz was insufficiently challenging, prompting the generation of more analytically demanding questions for the next session. The computed difficulty instruction is stored within the shared agent state and retrieved by the quiz generation agent during the subsequent assessment cycle.

2. Quiz Generation Agent: The quiz generation agent is responsible for producing assessment content grounded in the retrieved instructional material, subject to the difficulty level prescribed by the adaptive difficulty agent. A structured prompt is constructed incorporating the retrieved document context and the difficulty instruction, directing the language model to generate multiple-choice questions in which each item comprises four options and an accompanying brief explanation. The generated quiz is initially rendered in an editable state to facilitate HITL review and correction prior to session publication.

3. Review Agent: The review agent underwent substantial redesign relative to its mid-term prototype, in which validation functionality was rudimentary and offered no mechanism for identifying or correcting erroneous questions. The finalised implementation adopts a two-layer validation architecture. The first layer performs structural identification implemented entirely in Python, without invoking the language model. The generated quiz is evaluated against : no question must be empty, the correct answers is listed among the options, exactly four options per question should be listed. The second layer checks the generated quiz for semantic evaluation checking: clarity of quiz, options are plausible, unique questions and definitiveness of the correct answer. The quiz is rejected if the overall quality rating is poor and more than half of the questions are flagged by the reviewer.

4. Explanation Agent: The explanation agent is the final agent of multi agent pipeline. The approved quiz by the reviewer is then iterated to generate a concise explanation for the correct options such that it also help the students to understand the concepts. The explanation are visible to the students when they have submitted the quiz as a feedback alongside each question and answer pair.

C. Human-in-the-Loop Interface

Upon completion of the review agent's evaluation, the instructor is presented with the full question set in an editable interface. Questions that have passed the review process are displayed with a green border to indicate their validated status. Questions flagged by the semantic reviewer are displayed with a red border, accompanied by the reviewer's stated rationale and suggested correction. The instructor may modify the question text, alter any of the four answer options, reassign the correct answer via a dropdown selection, or delete the question entirely. Once all corrections have been applied and the instructor is satisfied with the content, a confirmation action triggers session creation. A regeneration option is also available, enabling the instructor to discard the current quiz entirely and initiate a new generation cycle from the beginning. After the approval of the instructor the quiz session is created, generated code can be communicated to the students so they can attempt the quiz.

D. Database Design

The system employs a single SQLite database file comprising two relational tables: the sessions table and the attempts table. The sessions table stores the session code as the primary key, along with the topic focus string, the complete quiz as a serialised JSON object, and a creation timestamp. The attempts table maintains a foreign key reference to the sessions table via the session code and additionally records the student's name, their numerical score, and the timestamp of submission. SQLite was selected as the database engine due to its suitability for single-instance, file-based deployments, its absence of server infrastructure requirements, and its sufficient read/write performance for the prototype's operational scale.

VI. RESULTS AND DISCUSSION

A. System Functionality Evaluation

Several tests on the system and tested in on three different study materials: deep learning, affective computing and machine learning. Each tests were complete from end to end like from the teacher panel uploading and generating the session code to attempting the quiz then as the instructor monitoring the performance and analytics. The structural format of the quiz was passes almost 92% every session. The failures was mostly from the LLM generating wrong options for the questions. Quiz generation time ranged from 18 to 35 seconds. The lesser time was because of the short documents and the documents large in size took the most time. HITL working was also checked and how functional was it as an added feature.
Table I summarizes the key quantitative observations recorded during prototype evaluation.

Metric	Observed Value
Questions generated per session	3-10
Structural review pass rate	~92 %
Semantic review flag rate	~8-15 %
Avg. Quiz generation time	18-40 sec

Table I - Prototype Evaluation Summary

B. Retrieval Grounded Quiz Reality

To test the RAG pipeline we different set of documents of different topics. For example when it was tested on affective computing week lectures the questions produced were from the material itself and the options taken were also believable. This confirmed that the RAG pipeline was doing what it was supposed to. The same results were observed for different topics like deep learning documents statistics books and semester study material. The RAG pipeline grounded the LLM to produce only context specific material.

C. Adaptive Assessment Behaviour

To confirm that the adaptive algorithm works fine it was tested it on the same document twice. In the first trial the score went below the lower threshold of 0.4 as a result the next trials quiz generated was much simpler and the scores also improved. In a separate sessions we tested the scores above the higher threshold to test the difficulty level in a hard manner of the algorithm. When we got the scores above 0.7 18 threshold the next quiz session were much harder and then the scores were between the 50-60% range confirming that the algorithm works and manages the adaptability well. The boundary condition tests we wrote also confirmed that the threshold logic is correct. If the average is 0.4 then next session quiz generated will be of balanced level and if the average if 0.7 then instead of balanced we test the analytical level.

D. Human-in-the-Loop Architecture

The HITL feature we added provided instructors the sense of control and transparency to improve their overall experience. The instructors were able to correct the mistakes made by the LLM while generating the quiz. They can correct the questions and the options as well. They can also regenerate the whole quiz can also change how the question sounds. The HITL also provides a color coded distinction between the questions flagged by the reviewer and the correct ones. The flagged question are present in a red border with the reason of why they are flagged. The correct question are marked by green border.

E. Student Feedback and Explanation Quality

The explanations generated by the explanation agent were consistently clear and educational. They not only provided with the correct answer but also allowing them with the conceptual clarity and can also be used to learn. We tested the explanations after attempting the quiz we realised that these can also be used to learn. They helped to understand the reasoning behind why they were wrong.

F. Analytics Dashboard

The analytics dashboard we designed was built in Plotly. This include interactive charts that show the performance of individual students as well as across session. We designed the plots to be color coded with having different colors representing different scores. Scores below 40% are marked with red scores between 40-70% are shown in slight yellow color and score above 70% are shown in green.

VII. ETHICAL CONSIDERATIONS AND RISK MANAGEMENT

The integration of Artificial Intelligence in educational assessment introduces important ethical considerations related to fairness, transparency, and data privacy. AI-generated quiz content must be monitored to ensure that assessment items remain unbiased, pedagogically appropriate, and aligned with instructional objectives. Since the system relies on student performance data for adaptive difficulty adjustment, secure handling and storage of session-level learning information is essential to maintain confidentiality and trust.

In addition, explainability mechanisms are incorporated to ensure that automated evaluation outcomes can be interpreted by both educators and learners. The system is designed to support human oversight, allowing instructors to review generated quizzes and assessment analytics before deployment. Potential risks related to model hallucination, over-automation of evaluation processes, and algorithmic bias are mitigated through retrieval-grounded content generation, modular agent validation, and controlled classroom deployment strategies.

VIII. LIMITATIONS

The biggest limitation of the current system is that adaptive difficulty works at class level. Every student in a same session gets the same difficulty for the next quiz while we are not considering their individual performance. A student who scored 85% gets the same quiz as one who scored 20%. Achieving this student based personalisation would require tracking every student across multiple sessions, which we identified as future work from the start.

Another limitation was that the FAISS index is rebuilt from scratch every time a PDF is uploaded. If a teacher uploads the same document in multiple sessions, the embeddings get recomputed every time costing us time and resources. Due to this latency was faced and computational resources were wasted.

Also the review agent uses the same LLM which was used to generate the quiz, so there is a risk of the reviewer being biased toward approving questions that match the way it generates the output. And the system has not been tested under concurrent multiple user conditions. It is a single-instance Streamlit application and would need infrastructure changes if wanted to deploy it to a real classroom scale.

IX. CONCLUSION AND FUTURE WORK

This paper presented an AI-driven adaptive quiz system that integrates multi-agent orchestration with retrieval-augmented knowledge grounding to support intelligent classroom assessment. By combining document-based quiz generation, adaptive reasoning workflows, and real-time performance analytics, the proposed framework demonstrates the feasibility of scalable and context-aware automated evaluation. The implementation of HITL has also made the system more controllable for the instructor to verify and cross-check the LLM reasoning for the generated quiz. The modular architecture enables explainable assessment processes while supporting dynamic adjustment of quiz complexity based on collective classroom performance.

The most impactful improvement from the system end would be to introduce per-student adaptive difficulty. Instead of tracking the average of the session it can target each student performance and then can target them as per their ability. This will make the system more centered towards improving the students performance.

On the engineering side, if system is able to persist the FAISS using the document based hashing then in the case of same documents getting uploaded then there is no need to recompute the embeddings this would help us to reduce recomputation cost and time.

A formal evaluation study with a real student cohort would be valuable to compare outcomes between classes using the adaptive system and another group with static quizzes this would provide actual evidence of the system effectiveness beyond the prototype-level testing which was carried out.

Other useful extensions include support for open-ended format instead of just MCQ. Implementing OCR support for scanned PDFs and the one more ability that can make it more complete is we can offer the quiz as exportable documents.

REFERENCES

1. P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in *Advances in Neural Information Processing Systems*, 2020.
2. D. Chen, A. Fisch, J. Weston, and A. Bordes, “Reading Wikipedia to Answer Open-Domain Questions,” in *Proc. ACL*, 2017.
3. Q. Wu et al., “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation,” arXiv:2308.08155, 2023.
4. Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023.
5. A. Arslan and J. Cruz, “Comparative Analysis of Retrieval-Augmented Generation, Fine-Tuning and Prompt Engineering,” *IEEE Access*, 2023.
6. S. Kurdi, H. Leo, and K. Parsia, “A Systematic Review of Automatic Question Generation,” *Int. J. Artif. Intell. Educ.*, 2020.
7. K. VanLehn, “The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems,” *Educational Psychologist*, 2011.
8. N. T. Heffernan and C. L. Heffernan, “The ASSISTments Ecosystem: Building a Platform that Brings Scientists and Teachers Together,” *Int. J. Artif. Intell. Educ.*, 2014.
9. P. Brusilovsky, “Adaptive Hypermedia,” *User Modeling and User-Adapted Interaction*, 2001.
10. R. S. J. D’Mello and A. Graesser, “Intelligent Tutoring Systems,” *IEEE Intelligent Systems*, 2013.
11. G. Siemens and R. Baker, “Learning Analytics and Educational Data Mining: Towards Communication and Collaboration,” in *Proc. LAK*, 2012.
12. W. Holmes, M. Bialik, and C. Fadel, *Artificial Intelligence in Education: Promise and Implications for Teaching and Learning*. Center for Curriculum Redesign, 2019.
13. T. Brown et al., “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, 2020.
14. H. Kasneci et al., “ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education,” *Learning and Individual Differences*, 2023.
15. M. Heilman and N. Smith, “Good Question Generation,” in *Proc. NAACL HLT*, 2010.